Partial Co-Training for Virtual Metrology

Cuong Nguyen, Xin Li, Ronald D. (Shawn) Blanton
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA, USA
{cuongn, xinli, rblanton}@andrew.cmu.edu

Xiang Li
Singapore Institute of Manufacturing Technology
Agency for Science, Technology and Research (A*STAR)
Singapore
xli@simtech.a-star.edu.sg

Abstract—Virtual metrology is an important tool for industrial automation. To accurately build regression models for virtual metrology, we consider semi-supervised learning where labeled data are expensive to collect, but unlabeled data are abundant. In such a scenario, due to the scarcity of labeled data, traditional single-view learning methods face the risk of overfitting. To address the overfitting issue, we develop a Partial Co-training framework, which is an extension of the original co-training approach by means of an undirected probabilistic graphical model. Unlike other co-training techniques, this model creates a partial view by shrinking the original feature space, and makes use of this partial-view to provide guidance information for improving the complete-view model. Our approach is validated with data from two manufacturing applications. The results indicate that a consistent and robust estimation is achievable with very limited labeled data.

Keywords—virtual metrology; semi-supervised learning; co-training; multi-view learning

I. INTRODUCTION

Virtual metrology is an important tool for industrial automation where a performance metric of interest is predicted by machine parameters and production-line sensor data without the need for expensive physical measurement. Towards this goal, an accurate regression model must be created by using historical data.

In many practical applications, data samples are described by multiple features, which can be naturally split into more than one subset, and each subset itself is sufficient for the learning task. A classic example is webpage classification [1], where each webpage can be described not only by its textual content, but also by the hyperlink pointing to it. Both descriptions can be used to correctly classify the webpage, thus they are considered as two views. In general, when each data sample is described by multiple features partitioned into different subsets, each subset of features is referred to as a particular view.

In this work, we consider the single-view semi-supervised regression problem, where labeled data are expensive and unlabeled data are abundant. To illustrate this setup, a good example is the manufacturing testing scenario: final products of a manufacturing line are examined by a quality test. Each product has a single-view description which includes a number of features. For each product, these features are measured easily during the production phase before testing. On the other hand, quality-related measurements (the label) can only be obtained after the testing has been completed. In many manufacturing applications, the testing operation are exhaustive, or even destructive if the product is a consumable such as ink cartridge.

Therefore, sample labeling is a costly operation, and only a small amount of samples can be provided with label.

Such a semi-supervised regression scenario creates a problem for conventional single-view machine learning methods such as linear regression [2] [3], Gaussian process regression [4], or mixture models [5]. Due to the small amount of labeled samples, overfitting may occur: error is minimized on training set, but the model does not generalize well on the testing set. To address the overfitting issue, multi-view learning introduces a framework that assigns each view to one modeling function and jointly optimizes all functions to maximize the overall learning accuracy [6].

Co-training is one of the popular multi-view algorithms that apply multi-view learning in a semi-supervised arrangement. It was introduced by Blum and Mitchell in 1998 for classification [1]. Given two views of the input data, the co-training algorithm iteratively maximizes their consensus upon unlabeled data. During the past two decades, co-training has gained significant interest, and many variants of the co-training algorithm have been developed for numerous tasks, including regression [6]-[14]. In 2000, Nigam and Ghani developed a new algorithm based on EM, namely Co-EM [7]. Instead of assigning hard labels, Co-EM iteratively determines the probabilistic labels for unlabeled samples. In 2005, Sindhwani et al. proposed the idea of co-regularization [8], where the mutual agreement between two views is maximized upon unlabeled samples, while the loss is minimized on labeled samples. Later on, Brefeld et al. proposed a co-regression algorithm that employs an optimization scheme similar to co-regularization, but the objective is defined differently [9]. In 2011, Yu et al. developed a graphical representation of co-training. They introduced Gaussian process priors to multiple views, and named their method Bayesian Co-training (BCT) [10].

However, all the existing co-training regression algorithms require a dataset with distinct views, i.e. the views are defined to be non-overlapping subsets of features [15]. This is practically not possible in many single-view applications, such as the manufacturing testing scenario that we described. There is a need for a new co-training algorithm which adapts to single-view datasets, and at the same time, works well with limited amount of labeled data.

To address this need, we propose a novel Partial Co-training algorithm (PCT) that generates two regression functions on a single-view setup. Specifically, the PCT algorithm creates a simple regression function with a few parameters and a more complicated regression function with many parameters. It then uses the simple function together with unlabeled data to help

This research is supported in full by A*STAR Singapore.

978-1-5090-6505-9/17/$31.00 ©2017 IEEE
avoid overfitting for the complicated function. Our experiment results on industrial data sets demonstrate that the proposed method requires significantly fewer labeled samples than the traditional least-squares regression method (up to 37.5% reduction) without diminishing accuracy.

The remaining parts of this paper is organized as follows. Section II provides a review of two background algorithms that are most related to this work. The details of the PCT algorithm are described in section III. View creation procedure is discussed in section IV. In section V, experiments and results on real industrial datasets are discussed. Finally, we conclude the paper in section VI.

II. RELATED WORKS

Two related works: linear regression by least-squares fitting and Co-training are revised in this section.

A. Linear Regression by Least-Squares Fitting

In this subsection, we review a conventional machine learning method for regression, which is the linear basis function model [16]. Consider a single-view setup with a dataset of labeled samples and features: \( X = \{ \phi^{(j)} \}_{j=1}^{M} \). In this setup, \( X \) is a \( L \times M \) matrix and \( \phi^{(j)} \) is an \( L \times 1 \) vector corresponding to the \( j \)-th feature. The regression target (label) is denoted as \( y \), which is an \( L \times 1 \) vector. The goal is to estimate a real-valued label, which is assumed to have the following linear form:

\[
y = \sum_{j=1}^{M} \alpha_j \phi^{(j)} + \epsilon = X \alpha + \epsilon \tag{1}
\]

Model coefficients \( \alpha = [\alpha_1, \alpha_2, \ldots, \alpha_M]^T \) represents a linear combination of features, which needs to be learned. Uncaptured error is assumed to be Gaussian noise with zero mean and a small variance: \( \epsilon \sim N(0, \sigma^2 I) \). Although the regression function is linear with respect to \( \alpha \), it is generally a non-linear function of raw input features. This is because the features \( \phi \) are obtained by applying various basis functions to raw features. These basis functions are usually non-linear transformations such as sigmoidal basis functions, Fourier basis functions, or radial basis functions.

The maximum likelihood estimation (MLE) of the linear regression coefficients leads to minimization of the sum-of-squares error:

\[
\min_{\alpha} \| y - X \alpha \|^2_2 \tag{2}
\]

The solution to this minimization problem is known as the least-squares (LS) solution, and is obtained via the following equation:

\[
\alpha_{LS} = (X^T X)^{-1} X^T y \tag{3}
\]

As the number of labeled samples decreases, such that \( L \approx M \), the model encounters overfitting problem. An easy way to avoid overfitting is to add to the optimization problem an L-2 regularization term:

\[
\min_{\alpha} \| y - X \alpha \|^2_2 + \lambda \| \alpha \|^2_2, \tag{4}
\]

where \( \lambda \) is the regularization coefficient, which can either be pre-set or chosen via cross validation. This method is known as regularized least-squares regression (Reg LS) [16].

B. Co-training

The co-training algorithm was initially designed for classification. Nevertheless, the idea has been successfully applied to regression tasks. Below is a review of the original co-training algorithm described by Blum and Mitchel [1].

Consider again the dataset \( X \) described in the previous subsection. Assume that it has two distinct views, so that each sample is \( x = (x_1, x_2) \). Let \( X_L \) denote the set of \( L \) labeled samples, and \( X_U \) denotes the set of \( U \) unlabeled samples. The goal is to learn a binary classification function \( f(x) \in \{0,1\} \). Given this arrangement, the co-training algorithm is described as follows:

**Algorithm 1. Co-training**

1. create a pool \( X_U \) of unlabeled samples by choosing \( U \) samples at random from \( X_U \)
2. for \( k \) iterations
3. use \( X_L \) to train a classifier \( h_1 \) that considers only the \( x_1 \) portion of \( x \)
4. use \( X_L \) to train a classifier \( h_2 \) that considers only the \( x_2 \) portion of \( x \)
5. allow \( h_1 \) to label \( p \) positive samples and \( n \) negative samples from \( X_U \)
6. allow \( h_2 \) to label \( p \) positive samples and \( n \) negative samples from \( X_U \)
7. add these self-labeled samples to \( X_L \)
8. randomly choose \( 2p + 2n \) samples from \( X_U \) to replenish \( X_U \)
9. end for

The way Algorithm 1 works is briefly explained as follows. The authors assumed a conditional independence assumption, by which, given the class label \( f(x) \), two views \( x_1 \) and \( x_2 \) are conditionally independent [17]. Under this assumption, any self-labeled sample provided by \( h_1 \) is a random labeled sample for \( h_2 \), and vice versa. This means that each classifier is providing independent labeled samples to help train the other. After a sufficiently large number of iterations, both classifiers “agree” on labeling the dataset, and gradually converge to the target classifier.

III. PROPOSED METHOD

In this section, we start by constructing a graphical model for the PCT method and then deriving the maximum-likelihood solution to estimate model parameters from data. Several implementation details are also discussed. Finally, we provide a summary of the proposed method.

A. Graphical Model

Consider again the linear regression scenario in (1). Let \( Z \equiv \{ \phi^{(j)} \}_{j=1}^{P} \); \( P < M \) be an \( N \times P \) matrix that consists of a subset of features in \( X \). We thereby use lowercase \( x \) to denote a sample (a row) in \( X \), and lowercase \( z \) to denote a sample from \( Z \). In this
manner, two overlapping views of the data are defined, with \( X \) being the \textit{complete view} and \( Z \) being the \textit{partial view}.

Suppose the dataset consists of \( L \) labeled samples and \( U \) unlabeled samples, we write \( X_L \) or \( Z_L \) to denote the labeled samples, for which the target value \( y \) is known. Similarly, \( X_U \) or \( Z_U \) denotes unlabeled samples, for which \( y \) is unknown. Fig. 1 visualizes the described two-view dataset.

Assume that the labeled samples and unlabeled samples are generated independently, then from (8) and (9), the joint likelihood of the entire dataset is:

\[
\begin{align*}
\text{pdf}(y, f_{1L}, f_{2L}|\alpha, \beta) & \propto \exp(-\lambda_1 \| y - f_{1L} \|^2_2) \\
\text{exp}(-\lambda_1 \| y - f_{2L} \|^2_2). \exp(-\lambda_3 \| f_{1L} - f_{2L} \|^2_2). \\
\text{exp}(-\lambda_4 \| f_{1U} - f_{2U} \|^2_2) \\
\end{align*}
\]

Similarly, for unlabeled data, the likelihood function is:

\[
\begin{align*}
\text{pdf}(f_{1U}, f_{2U}|\alpha, \beta) & \propto \exp(-\lambda_4 \| f_{1U} - f_{2U} \|^2_2) \\
\end{align*}
\]

where new parameters \( \{\lambda_i^U\}_{i=1}^4 \) are deterministic functions of the hyper-parameters \( \{\sigma^{2}_1, \sigma^{2}_2, \sigma^{2}_3\} \) (more details are provided in Appendix A).

Assume that the labeled samples and unlabeled samples are generated independently, then from (8) and (9), the joint likelihood of the entire dataset is:

\[
\begin{align*}
\text{pdf}(y, f_{1L}, f_{2L}, f_{1U}, f_{2U}|\alpha, \beta) & \propto \exp(-\lambda_1 \| y - f_{1L} \|^2_2). \exp(-\lambda_1 \| y - f_{2L} \|^2_2). \\
\text{exp}(-\lambda_4 \| f_{1U} - f_{2U} \|^2_2) \\
\end{align*}
\]

Maximizing the likelihood function in (10) is equivalent to minimizing the negative log-likelihood function:

\[
L(\alpha, \beta) = \lambda_1 \| y - X_L \|_2^2 + \lambda_2 \| y - Z_L \|_2^2 \\
+ \lambda_3 \| X_L - Z_L \|_2^2 + \lambda_4 \| X_U - Z_U \|_2^2 \\

\text{Subject to} \quad \alpha \geq 0, \quad \beta \geq 0
\]

Thus, MLE solution is achieved via solving the following optimization problem:

\[
(\alpha^*, \beta^*) = \underset{\alpha, \beta}{\text{arg min}} L(\alpha, \beta)
\]

From (12), we would like to make several remarks. Firstly, each of the first two terms resembles the classic linear regression as in (2). Secondly, the third and fourth term enforce the consistency between the two regression models upon labeled and unlabeled data. Thirdly, this is a weighted optimization, as in (2). Lastly, the objective function is quadratic and convex, thus an analytical solution can be found. Our derivation of the closed-form solution is given in Appendix B.

Let \( f_1(X) \) be a linear regression function on \( X \), in other words, \( f_1 \) is the \textit{complete model}. Similarly, let the \textit{partial model} \( f_2(Z) \) be a linear regression function on \( Z \):

\[
f_1(X) = X. \alpha, \quad f_2(Z) = Z. \beta
\]

Since the partial model has less parameters than the complete model, it is more robust against overfitting. If the partial model is sufficiently accurate, it can be used together with unlabeled data to provide pseudo-labeled samples to help train the complete model. In this way, we can prevent the complete model from being overfitted. This helpful relationship is utilized by including a \textit{consensus function} \( f_c \) between two models. Fig. 2 shows the undirected graphical model for two cases: labeled and unlabeled data. We assume a zero-mean Gaussian distribution over the difference between the consensus function and each regression function:

\[
f_c - f_1 \sim \mathcal{N}(0, \sigma^2_1 I), \quad f_c - f_2 \sim \mathcal{N}(0, \sigma^2_2 I)
\]

Finally, the measurement noise associated with label \( y \) measurement is parameterized by the following relationship:

\[
y - f_c \sim \mathcal{N}(0, \sigma^2 y)
\]

By the rules of undirected graphical model, the random variables \( y, f_1, \) and \( f_2 \) are conditionally independent of each other given \( f_c \). This assumption is analogous to the conditional independence assumption of Algorithm 1, where the two views are assumed to be conditionally independent given the label. At the functional level, the label \( y \) only depends on \( f_c \), and regression functions only depend on each other via the consensus function \( f_c \) [10].

The PCT graphical model is completely defined with parameters \( \{\alpha, \beta\} \) and hyper-parameters \( \{\sigma^2_1, \sigma^2_2, \sigma^2 y\} \). The learning goal is to estimate parameters \( \alpha \) and \( \beta \) from training data. This is accomplished via MLE, and is explained in the next section.

B. Maximum Likelihood Estimation

Using the probabilistic interactions defined in (6) and (7), the likelihood function for labeled data is:

\[
\begin{align*}
\text{pdf}(y, f_{1L}, f_{2L}|\alpha, \beta) & \propto \exp(-\lambda_1 \| y - f_{1L} \|^2_2). \\
\exp(-\lambda_1 \| y - f_{2L} \|^2_2). \exp(-\lambda_3 \| f_{1L} - f_{2L} \|^2_2). \\
\end{align*}
\]

Similarly, for unlabeled data, the likelihood function is:

\[
\begin{align*}
\text{pdf}(f_{1U}, f_{2U}|\alpha, \beta) & \propto \exp(-\lambda_4 \| f_{1U} - f_{2U} \|^2_2), \\
\end{align*}
\]

where new parameters \( \{\lambda_i^U\}_{i=1}^4 \) are deterministic functions of the hyper-parameters \( \{\sigma^{2}_1, \sigma^{2}_2, \sigma^{2}_3\} \) (more details are provided in Appendix A).

Assume that the labeled samples and unlabeled samples are generated independently, then from (8) and (9), the joint likelihood of the entire dataset is:

\[
\begin{align*}
\text{pdf}(y, f_{1L}, f_{2L}, f_{1U}, f_{2U}|\alpha, \beta) & \propto \exp(-\lambda_1 \| y - f_{1L} \|^2_2). \exp(-\lambda_1 \| y - f_{2L} \|^2_2). \\
\exp(-\lambda_4 \| f_{1U} - f_{2U} \|^2_2) \\
\end{align*}
\]

Maximizing the likelihood function in (10) is equivalent to minimizing the negative log-likelihood function:

\[
L(\alpha, \beta) = \lambda_1 \| y - X_L \|_2^2 + \lambda_2 \| y - Z_L \|_2^2 \\
+ \lambda_3 \| X_L - Z_L \|_2^2 + \lambda_4 \| X_U - Z_U \|_2^2 \\

\text{Subject to} \quad \alpha \geq 0, \quad \beta \geq 0
\]

Thus, MLE solution is achieved via solving the following optimization problem:

\[
(\alpha^*, \beta^*) = \underset{\alpha, \beta}{\text{arg min}} L(\alpha, \beta)
\]

C. Estimation of Hyper-parameters

The hyper-parameters \( \{\sigma^2_1, \sigma^2_2, \sigma^2 y\} \) need to be numerically specified before solving (12). A brute-force cross validation to determine the hyper-parameter values would involve scanning a three-dimensional table, which is computationally prohibitive. On this account, we suggest a heuristic-based estimation for each of them. Suppose \( f_c \) is the true underlying regression function, the variance \( \sigma^2 y \) between \( y \) and \( f_c \) speaks for

Fig. 1. Visualization of a dataset with complete view \( X \) and partial view \( Z \).

Fig. 2. PCT graphical model for (a) labeled data and (b) unlabeled data.
measurement error during label collection. In our experiments, we assume this measurement error is insignificant, so that the consensus function \( f_\text{c} \) is equal to the training label \( y \), i.e. \( \sigma_2^2 = 0 \). Since the partial model has low complexity, its variance \( \sigma_2^2 \) can be accurately estimated during training from a relatively small number of labeled samples. Accordingly, we make use of leave-one-out cross-validation [16] as follows:

**Algorithm 2. Leave-one-out Cross-validation**

Given \( L \) labeled samples \( \{(x_\text{t}, y_\text{t},)\}_{l=1}^{L} \), compute the leave-one-out mean-squared-error \( \xi_{Loo}^{(l)} \).

1. for \( i = 1, 2, ..., L \)
2. leave out \( (x_\text{t}, y_\text{t}) \) as temporary test sample
3. train a least-square partial model \( f_\text{c}(x) \) using remaining samples \( \{(x_\text{t}, y_\text{t})\}_{l=1}^{L} \setminus \{(x_\text{t}, y_\text{t})\} \)
4. \( \xi_i = y_\text{t} - f_\text{c}(x_\text{t}) \)
5. end for
6. \( \xi_{Loo} = \frac{1}{L} \sum_{i=1}^{L} \xi_i^2 \)

Using Algorithm 2, the estimator for \( \sigma_2^2 \) is \( \hat{\sigma}_2^2 = \xi_{Loo} \). Leave-one-out cross-validation is preferred over other types of cross-validation due to the scarcity of labeled data. The last hyper-parameter is \( \sigma_2^2 \) – variance of error between complete model \( f_\text{c} \) and consensus function \( f_\text{c} \). Since the complete model has high complexity, estimating \( \sigma_2^2 \) from limited training data is unreasonable. To address this issue, we again use leave-one-out validation. Given a set of \( V \) possible \( \sigma_2^2 \) values \( \Gamma = \{y_v\}_{v=1}^{V} \), we estimate the error of the PCT model associated with each \( v \) using leave-one-out cross validation, then choose the value that gives minimum validation error:

\[
\hat{\sigma}_1^2 = \arg\min_{\gamma \in \Gamma} \epsilon_{\text{PCT}}(y_v)
\]

(13)

It is worth noting that our heuristic-based estimations are not universally optimal, there may be other hyper-parameter estimation methods which are sufficiently as good.

**D. Partial Co-training Algorithm**

We summarize PCT with the following algorithm:

**Algorithm 3. Partial Co-training**

Given labeled samples \( \{(x_\text{t}, y_\text{t})\}_{l=1}^{L} \) and unlabeled samples \( \{(x_\text{t}, y_\text{t})\}_{l=1}^{L+u} \), estimate \( (\alpha_{\text{PCT}}, \beta_{\text{PCT}}) \)

1. \( \hat{\sigma}_2^2 = 0 \)
2. estimate \( \sigma_2^2 \) using Algorithm 2
3. generate a set of possible \( \sigma_2^2 : \Gamma = \{y_v\}_{v=1}^{V} \)
4. for \( v = 1, 2, ..., V \)
5. \( \sigma_1^2 = y_v \)
6. compute \( \{\lambda_i\}_{i=1}^{V} \) from the given \( (\sigma_1^2, \sigma_2^2, \sigma_3^2) \)
7. for \( i = 1, 2, ..., L \)
8. leave out \( (x_\text{t}, y_\text{t}) \) as temporary test sample
9. by solving (12), learn \( (\alpha^{(i)}, \beta^{(i)}) \) from remaining samples
10. \( \xi_i = y_\text{t} - x_\text{t}, \alpha^{(i)} \)

IV. VIEW CREATION

The creation of views is important to multi-view learning in general [6]. For PCT, it is essential to indicate how complete and partial views can be selected from a pool of available features.

Given a certain amount of data, we wish to find the largest possible complete view that does not overfit the training dataset. A model built upon such a complete view would have maximum representative capacity. On a side note, because a partial view contains features which are part of the complete view, one may generate numerous “candidate” partial views and pick the one with best validation error. However, the number of candidates grows exponentially with respect to the number of features in the complete view. Therefore, given a large pool of input features, we aim at completing two tasks: (i) reduce the number of features in the complete view, as long as the view contains sufficient information for regression; (ii) provide a relevant hierarchy of partial views that allows more efficient candidate examination.

**Sequential Forward Selection (SFS)** [18] adequately resolves both tasks. SFS works in a bottom-up stepwise manner. At each step, it scans the pool of features and includes the feature that leads to minimal validation error. Our SFS algorithm is described as follows:

**Algorithm 4. Sequential Forward Selection**

Given pool of \( F \) available features \( \mathcal{A} = \{\phi^{(j)}\}_{j=1}^{F} \), select a subset \( S \) that includes \( R \) features.

1. create an empty set \( S = \emptyset \)
2. for \( i = 1, 2, ..., R \)
3. for each \( \phi^{(j)} \in \mathcal{A} \)
4. generate a view \( S' = S \cup \{\phi^{(j)}\} \)
5. compute validation error \( \xi^{(j)} \) of \( S' \) using Algorithm 2
6. end for
7. \( \epsilon_{\text{SFS}}^{(i)} = \min_{\phi^{(j)} \in \mathcal{A}} \xi^{(j)} \)
8. \( \phi_{\text{SFS}}^{(i)} = \arg\min_{\phi^{(j)} \in \mathcal{A}} \xi^{(j)} \)
9. \( S = S \cup \{\phi_{\text{SFS}}^{(i)}\} \)
10. \( \mathcal{A} = \mathcal{A} \setminus \{\phi_{\text{SFS}}^{(i)}\} \)
11. end for

The complexity of SFS is \( O(FR) \), which is acceptable because this is a one-time execution.
To determine the complete model, consecutive errors $e^{(j)}_{SFS}$ are observed. Ideally, as more features are added, these consecutive errors should be monotonically decreasing. However, since the amount of training data is supposed to be small, there is a point $i^*$ at which the model complexity is too high, such that overfitting occurs. Overfitting causes $e^{(0)}_{SFS}$ to increase. Let $M$ denote the number of features in the complete view, then $M = i^* - 1$. We determine the complete view to be the subset of $S$ up to and including the $M^{th}$ feature: $X = \{ (1)_{SFS}, (2)_{SFS}, \ldots, (M)_{SFS} \}$.

Despite being sub-optimal, SFS generates an ordered feature set, which makes the subsequent step – partial view search – more efficient. The features in $X$ are being added in an order of descending importance. Hence, we generate $O(M)$ candidate partial views by decrementing the complete view following the order in $X$. Consequently, the number of candidates is no longer exponential in terms of $M$, making our experiments computationally feasible.

After determining the complete view as well as a few of candidate partial views, we use leave-one-out cross-validation to choose the best candidate. Applying Algorithm 3, the validation error associated with each candidate is computed. The candidate that minimizes this validation error is selected for testing. Remarkably, as a result of having overlapping views, our view creation scheme is distinctively different from other existing view creation methods [19]-[21]. This distinction also makes PCT incomparable to existing co-training methods.

V. NUMERICAL EXPERIMENTS

The proposed method is validated on two industrial datasets: gun-drilling and inkjet printing. Both datasets resemble the semi-supervised learning scenario as described in the introduction. Simply put, labeled data are scarce, unlabeled data are abundant, and the task is regression for an initially single-view dataset. On each dataset, an experiment is conducted and the results are reported.

A. Gun-drilling Dataset

Gun-drilling is a self-guiding, single flute, off-center and asymmetrical cutting operation that produces a deep, straight hole through a tough material [22]. The product of this machining operation is a hole with extremely high depth-to-diameter ratio (depth from 2 m to 6 m, diameter from 5 mm to 20 mm). After a hole is produced, its diameter is measured as a process quality indicator. However, diameter measurement is a time-consuming operation, which usually takes from 8 to 12 hours per hole. Thus, the task is to estimate the diameter from sensory signals (acoustic emission [23], force and vibration [24]) collected during the drilling operation.

1) View Creation

In this dataset, there are only $L = 12$ labeled samples and $U = 28$ unlabeled samples. Raw input data contains nine features. We applied quadratic basis functions on the raw features to obtain 54 features. After executing Algorithm 4, the SFS results are plotted in Fig. 3. After adding the fifth feature, the regression model is excessively complex, causing overfitting occurred which leads to the increased validation error.

According to our view creation scheme, the complete view for this dataset is $X = \{ \phi^{(1)}, \phi^{(15)}, \phi^{(20)}, \phi^{(49)}, \phi^{(34)} \}$. Three candidate partial views $Z_1, Z_2, Z_3$ are also generated, they contain the first two, three, and four features of $X$, respectively.

2) Experimental Results

In this experiment, we keep the amount of unlabeled data unchanged, while shrinking the labeled set. Starting from the initial 12 labeled sample, we will gradually remove labeled samples. At each value of $L$, a leave-one-out test is used to compute testing errors of three models: PCT, LS, and Reg LS the latter two are based on the complete view. Then, testing errors are averaged over $L$ iterations and reported.

![Fig. 3. SFS results on gun-drilling dataset.](image)

![Fig. 4. Experiment results on gun-drilling dataset.](image)

Since the complete view is chosen such that the validation error is minimized, the corresponding LS and Reg LS models are the optimal linear model among all linear models that might be trained using only labeled data. Thus, by testing the PCT model versus the optimal LS and Reg LS models, we would like to know whether or not PCT can further improve the complete
model with the help of a partial model and unlabeled data. The results are provided in TABLE I, and plotted in Fig. 4.

Experiment results show that the proposed model is less susceptible than the least-squares models to overfitting. In fact, PCT produces lower testing error for any value of \( L \). The proposed method performs much better when \( L \) is small. In particular, by decreasing \( L \) from nine to eight, the LS error increases rapidly by 0.188\%, the Reg LS error increases by 0.168\%, while the PCT error rate increases by only 0.0039\%.

B. Inkjet Printing Dataset

In the printing industry, page-yield of an inkjet cartridge is the number of “good pages” that can be printed with that particular cartridge [25]. Apparently, a technical description of a “good page” is specified by an industrial standard. Page-yield is widely used as one of the most important quality indicators for ink cartridges. Nevertheless, the actual test to acquire page-yield is destructive, because not only is the cartridge consumed, but the printer being used in the process is also diminished after printing thousands of pages. There is a need for a regression model that can approximate the number of pages from pre-testing sensory measurements to avoid the destructive test.

1) View Creation

In this dataset, there are four smaller datasets corresponding to four ink colors: Black, Cyan, Magenta, and Yellow. TABLE II presents the number of labeled and unlabeled samples in each dataset. Originally, each dataset has seven raw features. We applied quadratic basis functions and obtain 35 features in total. The SFS results for Black dataset is shown in Fig. 5. The complete model is

\[
X = \{ \phi^{(9)}, \phi^{(11)}, \phi^{(14)}, \phi^{(24)}, \phi^{(6)}, \phi^{(20)} \}
\]

Three candidate partial views \( Z_1, Z_2, Z_3 \) include the first four, five, and six features in \( X \), respectively. The same view creation process is carried out for the Cyan, Magenta, and Yellow datasets.

| TABLE II. INKJET PRINTING DATASETS |
|-----------------|------------|------------|------------|----------|
| Black | Cyan | Magenta | Yellow |         |
| \( L \) | 16 | 16 | 16 | 16 |
| \( U \) | 42 | 40 | 36 | 42 |

Fig. 5. SFS results on Black dataset.

2) Experimental Results

Similar to the gun-drilling application, we conduct the same experiment for each Inkjet Printing dataset. The results are provided in Fig. 6. For each value of \( L \), least-squares fitting produces higher error than that of PCT. Again, the outcome of experiments demonstrates how the proposed method easily copes with the lack of labeled data. For instance, for the Black dataset, number of training labeled samples can be reduced by up to 37.5\%, without significant degrading accuracy.
VI. CONCLUSION

In this paper, a novel Partial Co-training method is proposed for semi-supervised regression tasks where labeled data are scarce, and unlabeled data are plentiful. First, candidate pairs of complete view and partial view are generated using SFS. After that, hyper-parameters of the PCT graphical model are estimated from heuristics. Finally, the maximum-likelihood solution is found by solving a convex optimization problem. Justified by results from real industrial applications, the proposed method is robust against overfitting and performs reliably for a very limited amount of labeled data. Hence, it is of great value for many practical applications where obtaining a label is extremely expensive. In this paper, quality control in manufacturing serves as a good example of this scenario.

It is worth noting several aspects regarding the PCT model. Firstly, the success of PCT depends on whether the partial model is able to provide guidance information to boost the complete model. If so, the risk of overfitting on the complete model is substantially reduced. Secondly, this model introduces and suggests the beneficial employment of overlapping views in multi-view learning. Thirdly, the PCT graphical model is sufficient to serve as a modular unit or a supportive unit in a bigger system, thus making this work highly extensible.

ACKNOWLEDGMENT

This work is sponsored in full by A*STAR. The authors would like to thank Professor C. Fred Higgs, III for the valuable discussions on insights on gun-drilling.

REFERENCES


APPENDIX A

Given the PCT graphical model with parameters $\{\alpha, \beta\}$, it is necessary to integrate $f_c$ out of the joint density distribution of the graph, because $f_c$ is a latent variable. Consider a more general graphical model, where there are $m$ nodes corresponding to $m$ regression functions $\{f_j\}_{j=1}^m$. These nodes are not directly related, each of them is only connected to a consensus function $f_c$ via a Gaussian potential: $f_c - f_j \sim \mathcal{N}(0, \sigma_j^2 I)$. The joint density distribution function of the graph is as follows:

$$f_{\text{joint}} = \prod_{j=1}^m f_j^{\alpha_j} f_c^{\beta_c} \prod_{j \neq k} f_{j,k}^{\alpha_{j,k}}$$

where $f_{j,k}$ is the connection between nodes $j$ and $k$. This distribution can be further simplified for the case where $f_{j,k}$ is a fully connected graph with $m$ nodes:

$$f_{\text{joint}} = \prod_{j=1}^m f_j^{\alpha_j} f_c^{\beta_c} \prod_{j \neq k} f_{j,k}^{\alpha_{j,k}}$$

Note that the $\alpha_{j,k}$ parameters are responsible for the weights of the connections between the nodes.
where $Z$ is a normalizing constant, and $a, b, c$ are given by: $a = \left( \sum_j \frac{1}{\sigma_j^2} \right), b = \sum_j \frac{f_j}{\sigma_j^2}, c = -\sum_j \frac{f_j^2}{2\sigma_j^2}$. Notice that $a, b, c$ are constants w.r.t. $f_c$. Integrating $f_c$ out of the joint distribution yields:

$$
\text{pdf}(f_1, ..., f_m|\alpha, \beta) = \int \text{pdf}(f_c, f_1, ..., f_m|\alpha, \beta) \, df_c \\
= \frac{1}{Z'} \exp \left( \frac{b^T b}{2a} + c \right) \\
= \frac{1}{Z'} \exp \left( \frac{1}{2a} \left( -\sum_j \sum_{k>j} f_j^2 f_k^2 - 2f_j f_k f_k + f_k^2 f_k \right) \right) \\
= \frac{1}{Z'} \exp \left( \frac{1}{2a} \left( \sum_j f_j^2 - 2f_j f_k + f_k^2 \right) \right) \\
= \frac{1}{Z'} \exp \left( -\sum_j \sum_{k>j} \lambda_{j,k} \| f_j - f_k \|^2 \right),
$$

in which, $Z'$ is another normalizing constant, while $\lambda_{j,k} = \left( 2a \sigma_j^2 \sigma_k^2 \right)^{-1} = \left( 2a \sigma_j^2 \sigma_k^2 \left( \sum_j \frac{1}{\sigma_j^2} \right)^{-1} \right)$. The PCT graphical model is actually a special case of this general model, thus $\{\lambda_{i}\}_{i=1}$ in (10) are expressed in terms of the hyper-parameters $\{\sigma_1^2, \sigma_2^2, \sigma_3^2\}$ as follows:

$$
\lambda_1 = \frac{1}{2} \left( \sigma_1^2 + \sigma_2^2 + \sigma_3^2 \right)^{-1}, \lambda_2 = \frac{1}{2} \left( \sigma_2^2 + \sigma_3^2 + \sigma_3^2 \right)^{-1}, \lambda_3 = \frac{1}{2} \left( \sigma_1^2 + \sigma_2^2 + \sigma_3^2 \right)^{-1}, \lambda_4 = \frac{1}{2} \left( \sigma_1^2 + \sigma_2^2 \right)^{-1}.
$$

**APPENDIX B**

In order to derive an analytical solution ($\alpha^*, \beta^*$) to the minimization problem in (12), we start by rewriting the objective function:

\[
\mathcal{L}(\alpha, \beta) = \lambda_1 \| y - X_L \alpha \|_2^2 + \lambda_2 \| y - Z_L \beta \|_2^2 + \lambda_3 \| X_L \alpha - Z_L \beta \|_2^2 + \lambda_4 \| X_U \alpha - Z_U \beta \|_2^2 \\
= \alpha^T A \alpha + \beta^T B \beta + \alpha^T P \alpha + \beta^T V + \beta^T W,
\]

where we define: $A = (\lambda_1 + \lambda_3)X_L^T X_L + \lambda_4 X_T^T X_U, B = (\lambda_2 + \lambda_3)Z_L^T Z_L + \lambda_4 Z_U^T Z_U, P = -2\lambda_3 X_L^T Z_L - 2\lambda_4 X_U^T Z_U, V = -2\lambda_4 X_L^T y, \text{ and } W = -2\lambda_2 Z_L^T y$. Since $\mathcal{L}(\alpha, \beta)$ is quadratic and convex, the optimal point ($\alpha^*, \beta^*$) satisfies:

\[
\begin{align*}
\frac{\partial \mathcal{L}(\alpha, \beta)}{\partial \alpha} &= 0, \quad \Rightarrow \quad 2A \alpha^* + P \beta^* + V = 0 \\
\frac{\partial \mathcal{L}(\alpha, \beta)}{\partial \beta} &= 0, \quad \Rightarrow \quad 2B \beta^* + P \alpha^* + W = 0
\end{align*}
\]

Hence, the analytical solution is:

$$
\alpha^* = \left( 2A - \frac{1}{2} PB^{-1}P^T \right)^{-1} \left( \frac{1}{2} PB^{-1}W - V \right), \beta^* = \left( 2B - \frac{1}{2} P^TA^{-1}P \right)^{-1} \left( \frac{1}{2} P^TA^{-1}V - W \right).
$$